Title	Efficient packet routing strategy in complex networks
Author(s)	Kawamoto, Hiroki; Igarashi, Akito
Citation	Physica A: Statistical Mechanics and its Applications (2012), 391(3): 895-904
Issue Date	2012-02
URL	http://hdl.handle.net/2433/151753
Right	© 2011 Elsevier B.V.
Type	Journal Article
Textversion	author

Efficient packet routing strategy in complex networks

Hiroki Kawamoto and Akito Igarashi

Department of Applied Mathematics and Physics, Kyoto University, Kyoto 606-8501 Japan

Abstract

We investigate a new efficient packet routing strategy which mitigates traffic congestion on complex networks. In order to avoid congestion, we minimize the maximum betweenness, which is a measure for concentration of routing paths passing through a node in the network. Danila *et al.* propose a packet routing strategy in which, instead of shortest paths, they used efficient paths, which are the paths with the minimum total summations of weights assigned to nodes in the respective paths. They use a heuristic algorithm in which the weights are updated step by step by using the information of betweenness of each node in every step and the respective total summations of weights for paths through the nodes with large degrees become comparatively large. Thus passage through such nodes, where congestion almost occurs, is likely to be avoided in their algorithm. The convergence time by their algorithm is, however, quite long. In this paper, we propose a new efficient heuristic algorithm which balances traffic on networks by achieving minimization of the maximum betweenness in the much smaller number of iteration steps for convergence than that by the algorithm of Danila *et al.*

Keywords: Packet routing, Complex network, the Internet

1. Introduction

Since the discovery of small world networks, where average lengths of shortest paths, which have the respective minimum numbers of hops between two nodes, are proportional to the logarithm of the total number of nodes, by Watts and Storogatz[1], and scale-free networks, where the distribution of degree k follows $P(k) \sim k^{-\gamma}$, by Barabási and Albert[2], the structures and dynamics on the networks have been investigated actively in many fields. The small world and scale-free networks are called complex networks[3].

Recently, communication networks such as the Internet reach a huge scale and a lot of data pass through the networks. Although some attempts to optimize information traffic are made in real computer networks, they are not systematic. Therefore, development of efficient packet routing strategies optimal for transport is one of the most important problems in the study of computer networks. Until now, however, transport routing strategies used on computer networks have been solely determined based on the shortest paths. Real computer networks, however, display scale-free features and have poor performance for traffic flow on routing based on the shortest path strategy (SPS). The reason is because traffic concentrates on hubs, nodes with a great many degrees, and congestion is prone to occur there when we send traffic by the SPS because many shortest paths pass through hubs on scale-free networks [4]. It is especially important that we find the routing strategy which can bear as much traffic as possible without congestion in scale-free networks. Thus it is necessary to devise routing strategies to avoid hubs efficiently in place of those based on the SPS.

Many studies on information transport on complex networks have attracted a great deal of interest in recent years[4–19]. Krause *et al.*[13–16] investigate routing in wireless ad hoc communication.

In this paper, we investigate packet routing strategies to mitigate traffic congestion in complex networks. In order to avoid congestion, nodes passed through by a great many routing paths should be bypassed. Concentration of routing paths to a node is measured with betweenness, which is calculated with the fast algorithm by Newman[20], and Newman and Girvan[21]. If we minimize the maximum betweenness, which is the largest one of all the nodes in the network, with an improvement of the routing strategy, congestion is reduced. Danila *et al.* propose a packet routing strategy in which, instead of shortest paths, they used efficient paths, which are the paths with the minimum total summations of weights assigned to nodes in the respective paths. They investigate a heuristic iteration algorithm which improves tolerance to congestion step by step by an increase of the weight for the node with the maximum betweenness in every step[5, 6] and the respective total summation of weights for paths through hubs become comparatively large. Thus routing properties were improved on congestion with the use of efficient paths, since, with the strategy, passage through nodes with hubs on scale-free networks is likely to be avoided. Hereafter, the algorithm proposed by Danila *et al.* is called the optimal path strategy (OPS). The convergence time of the OPS is, however, quite long. We propose a new heuristic algorithm which

converges quickly. That is, in a step of iteration, we change simultaneously the weights of several nodes with comparatively large betweennesses in the early stage and change only the weights of nodes with betweennesses which are exceedingly near to the maximum betweenness in the late stage and demonstrate that the algorithm constructs the routing paths which greatly mitigate congestion by minimizing the maximum betweenness in the much smaller number of iteration steps for convergence than that by the OPS.

The outline of this paper is as follows. In Section 2, we explain the relation between betweenness and congestion. In Section 3, we define the efficient paths. In Section 4, we consider the new heuristic algorithm renewing the next step strategy by using the information of the betweenness in every step and solve the problem of slow convergence by the existing algorithm. We also show the efficiency of the routing strategy from numerical simulation. In Section 5, we summarize our results and state some conclusions.

2. Betweenness and congestion

First, we explain the definition of betweenness, a packet routing model used in this paper, the relation between betweenness and congestion on the packet routing model and define a congestion index.

2.1. Betweenness

We consider that paths for all pairs of nodes in the network are defined according to a rule, for example, choosing the shortest path and so on. The betweenness of a node v according to the rule is defined as follows[20]:

$$b(v) = \frac{1}{N(N-1)} \sum_{s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{1}$$

where N denotes the total number of nodes in the network, $\sigma_{st}(v)$ the number of paths from a source node s to a target node t that pass through v and σ_{st} the number of all paths from s to t according to the rule, that is, $\sum_v \sigma_{st}(v)$. We normalize this index by $N(N-1)$, the total number of all pairs of a destination node and source node in the whole network. In other words, betweenness $b(v)$ means the probability of paths passing through v in all paths which are defined according to the rule.

2.2. Packet routing model

In this paper, we regard all nodes as both hosts and routers. That is, nodes generate and deliver packets. The data is divided into packets, which are sent to a destination through networks. Precisely, we assume a packet routing model as follows:

- The network is constructed with the Barabási and Albert model.[2]
- A packet is generated at every node at the same average rate λ per time step.
- The destination of the generated packet is determined at random.
- All nodes can deliver at most D packets to directly connected nodes in one step.
- A packet is removed from the network if the packet arrives at its destination.
- Each node has a First-In-First-Out (FIFO) queue for packets.
- The maximum length of the queue is infinite and we do not consider discard of packets when the queue is full.
- Each link is undirected.

2.3. The relation between congestion and betweenness

The critical rate of packet generation $\lambda_{\rm c}$ is defined as follows: when the rate λ of packet generation is less than $\lambda_{\rm c}$, the number of packets generated at source nodes and arriving at target nodes are balanced and packets flow freely. On the other hand, traffic congestion occurs as the number of packets inserted in the network exceeds that of removed packets per unit step and the number of heaped packets increases with time for $\lambda > \lambda_{\rm c}$. Thus. we define $\lambda_{\rm c}$ at which the phase transition takes place from free flow to congested traffic. This critical value can reflect the maximum capability of a system handling its traffic.

$B_{\rm max}$ is defined as the maximum betweenness of all nodes in the whole network and L as the average path length in the network. We estimate $\lambda_{\rm c}$ by using $B_{\rm max}$ as follows: In the free flow state, the total number of packets in the network is estimated to be λNL and the probability that a packet

exists at the node with $B_{\max}$ is approximated with $B_{\max}/L$, since on average the positions of the packets are distributed homogeneously in a routing path and the concentration of routing paths passing through the node is measured with betweenness. Therefore the maximum number of the packets sent at the node per step is estimated to be

$$\lambda N L \times \frac{B_{\max}}{L} = \lambda N B_{\max}. \tag{2}$$

Because a phase transition takes place from the free flow state to congestion when the number of sent packets exceeds D, $\lambda_{\rm c}$ is determined as follows [10]:

$$\lambda_c N B_{\max} = D \to \lambda_c = \frac{D}{N B_{\max}} \tag{3}$$

From this fact, it is found that we should minimize $B_{\max}$ in order to maximize $\lambda_{\rm c}$ and avoid congestion, if we fix the number of the delivering packet in one step at each node to be D, for all nodes. For this purpose delivery through the efficient paths, explained in the next section, is employed as the rule for packet routing in place of delivery through the shortest paths. Hereafter, we set $D = 1$ in this paper.

2.4. Congestion index

We define a congestion index $\eta(\lambda)$ to represent a phase transition from free flow to congested traffic as follows:

$$\eta(\lambda) = \lim_{\Delta t \to \infty} \frac{1}{N\lambda} \frac{\langle \Delta W \rangle}{\Delta t}, \tag{4}$$

where $\Delta W = W(t + \Delta t) - W(t)$, $W(t)$ is the total number of packets at step t in the network, $\langle \cdots \rangle$ denotes the average and $\langle \Delta W \rangle$ is divided by $N\lambda\Delta t$, which is the average number of packets generated in Δt steps. $\eta(\lambda)$ is zero in free flow because the number of generated packets is balanced with that of packets removed due to arrival at the destination. On the other hand, $\eta(\lambda)$ becomes positive in congested traffic because the total number of packets increases in the network. Therefore λ where $\eta(\lambda)$ becomes positive from zero is regarded as the critical generation rate λ_c from numerical simulation.

3. Efficient path

Since it is known that the router networks are scale-free networks [2], it is necessary to contrive new strategies which take the place of the SPS because a great many shortest paths concentrate in hubs and it causes traffic congestion in scale-free networks. We explain efficient paths which mitigate traffic congestion on scale-free networks.

A path $P[i \to j]$ from node i to node j with successive hops from i to x_1, $\cdots$, x_n, $\cdots$ and j, $(i \to x_1 \to \cdots \to x_n \to \cdots \to j)$, is defined as follows:

$$P[i \to j] \equiv (i \to x_1 \to \cdots \to x_n \to \cdots \to j). \tag{5}$$

We assign a weight on every node and the total weight of the path $S(P[i \to j])$ is defined as follows:

$$S(P[i \to j]) = \sum_m W_{x_m}, \tag{6}$$

where W_h is the weight of node h and m varies over nodes in the path $P[i \to j]$. The efficient path $\Gamma[i \to j]$ is defined by minimization of the total weight of the path as follows:

$$\Gamma[i \to j] \equiv \arg\min_{P[i \to j]} S(P[i \to j]). \tag{7}$$

$\Gamma[i \to j]$ always corresponds to the shortest path when the weights of all nodes in the network are equal to a constant.

There is a strong positive correlation between the degree and betweenness of a node with the SPS in scale-free networks[17]. Therefore traffic concentrates at hubs and traffic congestion is usually generated there if packets are sent through the shortest paths. For avoidance of passage through high degree nodes and congestion, the efficient paths with the weight $W_h = k_h^\alpha$, where k_h is the degree of node h and α a constant, are proposed for packet routing[17]. This strategy composes routes with the reduction of traffic concentration by avoidance of passage through high-degree nodes, since, for positive α, nodes with high degrees are likely to be avoided by minimization of S. Hereafter, we call the strategy the degree efficient path strategy (DEPS).

In the next section, we improve the OPS, the heuristic algorithm for determining optimal weights of nodes in the network proposed by Danila *et al.* [5, 6] who renew step by step tolerance to congestion by using the information of the betweenness in every step.

4. Heuristic algorithm

The problem of finding the exact optimal (smallest) maximum betweenness $B^*_{\max}$ for the efficient paths is mathematically equivalent to the problem of minimizing the sparsity vertex separator[18] which has been shown to be an NP-hard problem.[22] Thus, it is generally hard to find the exact solution of this problem within practical calculation times. First, the OPS is explained.

4.1. The algorithm of Danila et al., the OPS

It is shown that a near-optimal solution for $B^*_{\max}$ is found by the OPS. The algorithm proceeds as follows:[5, 6]

1. Assign weight 2 to every link and compute the "efficient" paths, where the total summation of the weights of the links in each path is minimum between all pairs of nodes in the initial condition. Since initially the weights of all links are equal, the initial efficient paths coincide exactly with the shortest paths.
2. Compute the betweenness of every node for the efficient paths.
3. At iteration step t, find the node which has the highest betweenness $B_{\max}(t)$ and add 1 to the weight of every link that directly connects it to another node.
4. Recompute the efficient path.
5. $t = t + 1$ and repeat 2 to 4.

This algorithm is proven to be equivalent to the following algorithm.

1. Assign weight 1 to every node and compute the efficient paths defined in Sec.3 between all pairs of nodes in the initial condition. Initially the determined paths coincide exactly with the shortest paths, because all weights of the nodes are the same.
2. Compute the betweenness of every node.
3. At iteration step t, find the node which has the highest betweenness $B_{\max}(t)$ and add 1 to the weight of this node.
4. Recompute the efficient path.
5. $t = t + 1$ and repeat 2 to 4.

Although a very small and near-optimal limit value of $B^*_{\max}$ is obtained, it takes a great many steps for convergence of $B_{\max}(t)$ relative to the system size by the algorithm.

4.2. Improvements of the OPS

Our goal is to reduce convergence time because it takes a long time to find the optimal solution by the OPS. There are two points in the algorithm which should be improved to reduce convergence time, as follows:

1. At every step, the weights of the links directly connected to the node with the maximum betweenness are only increased, or the weight of the node with the maximum betweenness is only increased in the equivalent algorithm. Therefore, convergence speed is slow.
2. The initial maximum betweenness $B_{\max}(0)$ is large because the SPS is used in the initial condition.

We address these two problems in the OPS-equivalent algorithm as follows:

1. The weights for several nodes with comparatively large betweenness are simultaneously renewed at each iteration step.
2. The DEPS is used in the initial condition and $B_{\max}(0)$ is exceedingly reduced in advance in comparison with the OPS.

We propose new heuristic algorithms by taking into account these improvements after this. In this subsection, we set the total number of nodes $N = 500$ and the average degree $\langle k \rangle = 8$.

4.2.1. New algorithms 1 and 2

We define the weight of a node i as $W_i(t)$ in iteration step t and the updating rule is as follows:

1. Assign the initial weight $W_i(0) = k_i^{\alpha}$ to every node i.
2. Update the weight of node j which satisfies $B_j(t)/B_{\max}(t) > \ell$ as follows:

$$W_j(t+1) = W_j(t) + \frac{B_j(t)}{B_{\max}(t)}, \tag{8}$$

where ℓ is a positive constant, k_i the degree of node i and α a constant. We set α to be 0.7, since for the BA model we calculate $B_{\max}(0)$ with $N = 500$ and $\langle k \rangle = 8$ for various α $(0 < \alpha < 1)$ and find that $B_{\max}(0)$ is a minimum for $\alpha = 0.7$. We call it algorithm 1, hereafter.

In Fig.1, we show $B_{\max}(t)$ in using this algorithm for $\ell = 0.9$, 0.99 and 0.999. From this figure, we find that the limit value of $B_{\max}(t)$ is high for $\ell =$

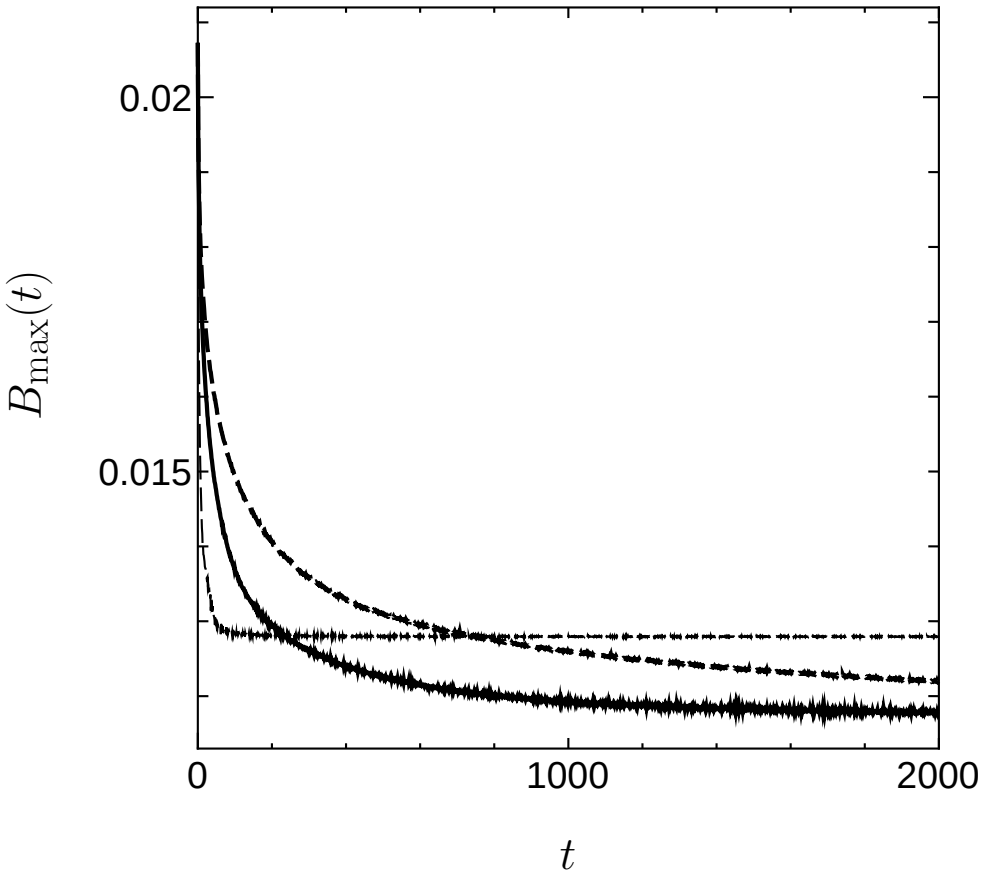

Fig. 1: $B_{\max}(t)$ with algorithm 1 using Eq.(8), averaged over 100 samples. Although, for $\ell = 0.9$, $B_{\max}(t)$ (the thin dashed line) converges fast, the limit value ≈ 0.013 is high. For $\ell = 0.99$ (the thick solid line), the limit value is sufficiently low. For $\ell = 0.999$, the limit value for quite large t is approximately coincident with that for $\ell = 0.99$. However, it takes a very long time for convergence for $\ell = 0.999$. (the thick dashed line) Therefore we take $\ell = 0.99$, because $B_{\max}(t)$ converges relatively fast to a satisfiable value.

0.9 and the number of iterations required to reach the convergence increases for $\ell = 0.999$, although the limit values for $\ell = 0.999$ are similar to those for $\ell = 0.99$. Therefore we conclude that this algorithm is optimal for $\ell = 0.99$.

In algorithm 1, we limit updates of weights to nodes with quite large betweenness using the parameter ℓ. Instead of ℓ, another parameter may also be used for the limitation. We define the updating of weights in algorithm 2 as follows:

1. Assign the initial weight $W_i(0) = k_i^\alpha$ to every node i with $\alpha = 0.7$.
2. Update the weight of node j as follows:

$$W_j(t+1) = W_j(t) + \left(\frac{B_j(t)}{B_{\max}(t)}\right)^a, \tag{9}$$

where a is a positive constant.

Since, for large a, the increment of the weights of the nodes with small betweenness is quite small, the updated amount of the weights of the nodes is actually limited. We show $B_{\max}(t)$ at step t for $a = 7$, 100 and 300 in Fig.2. From the figure, it is found that this algorithm is optimal for $a = 100$, since for $a = 7$ the limit value is not sufficiently low and for $a = 300$ convergence is much slower than for $a = 100$, where the limit value is a sufficiently low value which is similar to that for $a = 300$.

We show $B_{\max}(t)$ to compare algorithms 1 and 2 in Fig.3. From this figure, it is found that algorithm 2 is more efficient than algorithm 1 because the number of iterations required to reach convergence with algorithm 2 is less than that with algorithm 1, and the limit values with both algorithms are similar.

4.2.2. Improvement of algorithm 2

The number of iterations required to reach convergence is small at small a since the weights of many nodes are updated simultaneously in algorithm 2. Since $B_{\max}(t)$ becomes small and the number of nodes with the betweenness near $B_{\max}(t)$ increases as time elapses[8], the weights of many nodes are updated simultaneously for small a. With large a, we can carefully limit updates of weights of nodes to that of the node with quite large betweenness and the limit value of $B_{\max}(t)$ becomes sufficiently low since limited weights are slightly updated in a step. The convergence time for large a is, however, quite long. Accordingly, it is desirable to use small a at early steps and large

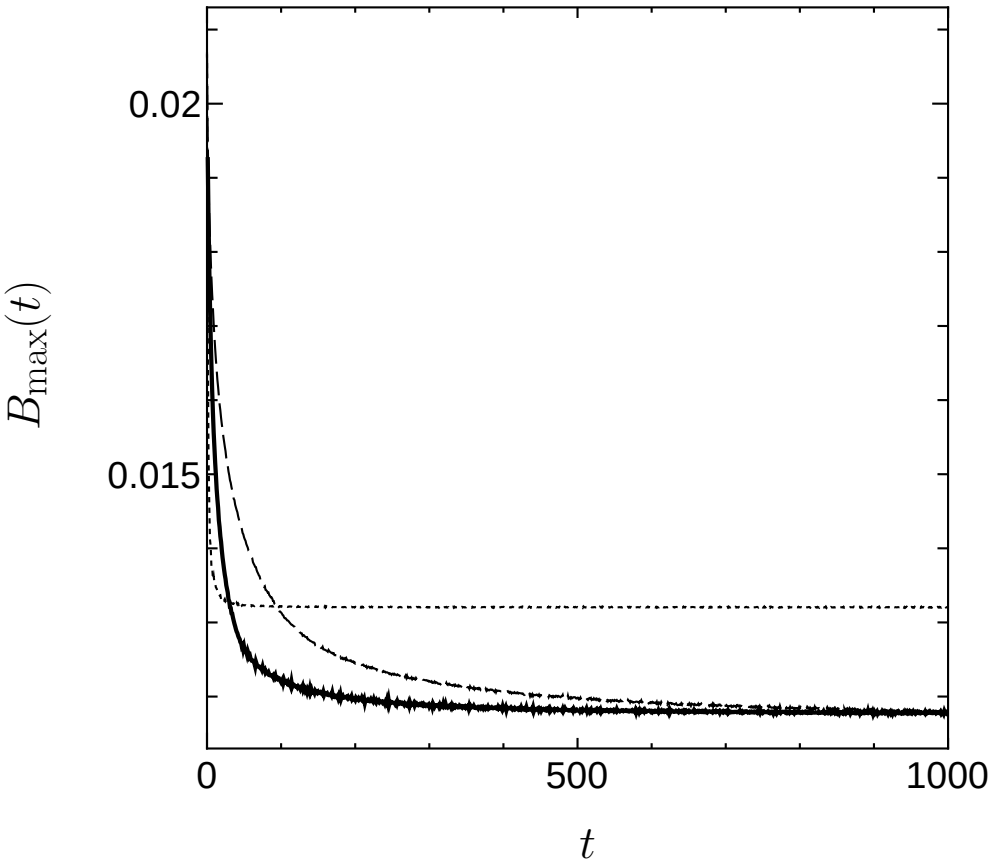

Fig. 2: $B_{\max}(t)$ is plotted as a function of t for $a = 7$ (dotted line), $a = 100$ (solid line) and $a = 300$ (dashed line) with the use of algorithm 2, averaged over 100 samples. Although for $a = 7$, $B_{\max}(t)$ converges faster than that for $a = 100$, the limit value for $a = 7$ is not sufficiently low in comparison with that for $a = 100$. The limit value for $a = 100$ is a sufficiently low value which is similar to that for $a = 300$. The convergence for $a = 100$ is faster than that for $a = 300$.

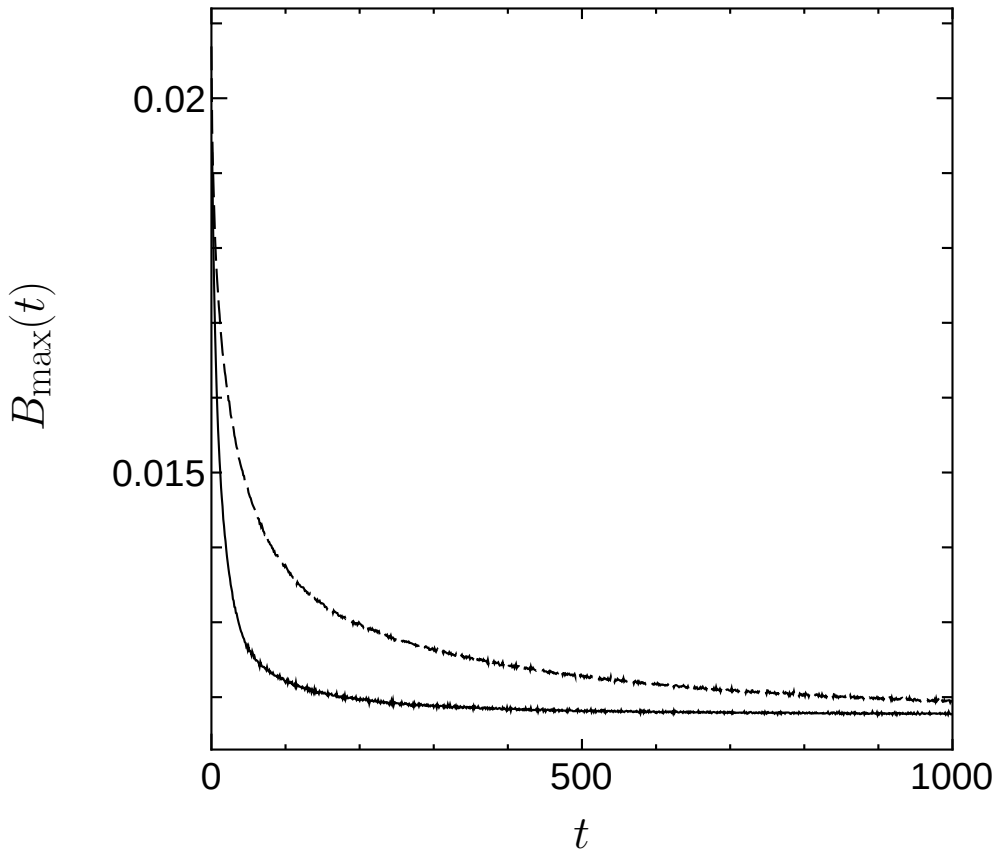

Fig. 3: The results for $B_{\max}(t)$ with algorithm 1 with $\ell = 0.99$ (dashed line) and algorithm 2 ($a = 100$, solid line) averaged over 100 samples are compared. We find that algorithm 2 is more efficient for fast convergence to a sufficiently low value than algorithm 1.

a at late steps, since at the early stage it must not be too careful to update the weights in order that we pursue the fast convergence and at the late stage a detailed update is needed for the optimal limit value of $B_{\max}(t)$ on the other hand. Therefore, we propose the algorithm with increasing a in Eq.(9) to take into account the above points.

We define algorithm 3 by considering this concept as follows:

1. Assign the initial weight $W_i(0) = k_i^{\alpha}$ to every node i for $\alpha = 0.7$.
2. Update the weight of node j as follows:

$$W_j(t+1) = W_j(t) + \left(\frac{B_j(t)}{B_{\max}(t)}\right)^{a(t)}. \tag{10}$$

Here, $a(t)$ is a monotonically increasing function of t. Although we try to use various functions for $a(t)$, the results with such functions are much the same as those with $a(t) = A + t$, where A is a constant. Consequently, we use $a(t) = A + t$ for $A = 5$.

Now, we show $B_{\max}(t)$ at step t by algorithms 2 and 3 ($a(t) = 5 + t$) in Fig.4. From this figure, it is found that the number of iterations required to reach convergence is greatly improved in algorithm 3 in comparison with algorithm 2 and that the limit values with both algorithms are similar. Thus, algorithm 3 is the most efficient for packet routing. We call algorithm 3 ($a(t) = 5 + t$) the new optimal path strategy (NOPS) hereafter.

Lastly, we compare the NOPS with the OPS in Fig.5. From the figure, with the NOPS we can amazingly decrease the number of iterations required to reach the limit while the sufficiently low limit value similar to that with the OPS is obtained.

4.3. Stop condition

We define the stop condition of the NOPS as follows:

1. $(B_{\max}(t))_{\min}$ is defined as the minimum of all $B_{\max}(t')$ for $t' \leq t$, that is, $\min_{t' \leq t} B_{\max}(t')$.
2. At every iteration step t, $B_{\max}(t)$ is compared with $(B_{\max}(t))_{\min}$. If $B_{\max}(t)$ is less than $(B_{\max}(t))_{\min}$, we set $(B_{\max}(t))_{\min} = B_{\max}(t)$ and record the weight of every node at this step in order to memorize the optimal weights of all nodes which are used for packet routing on the network.

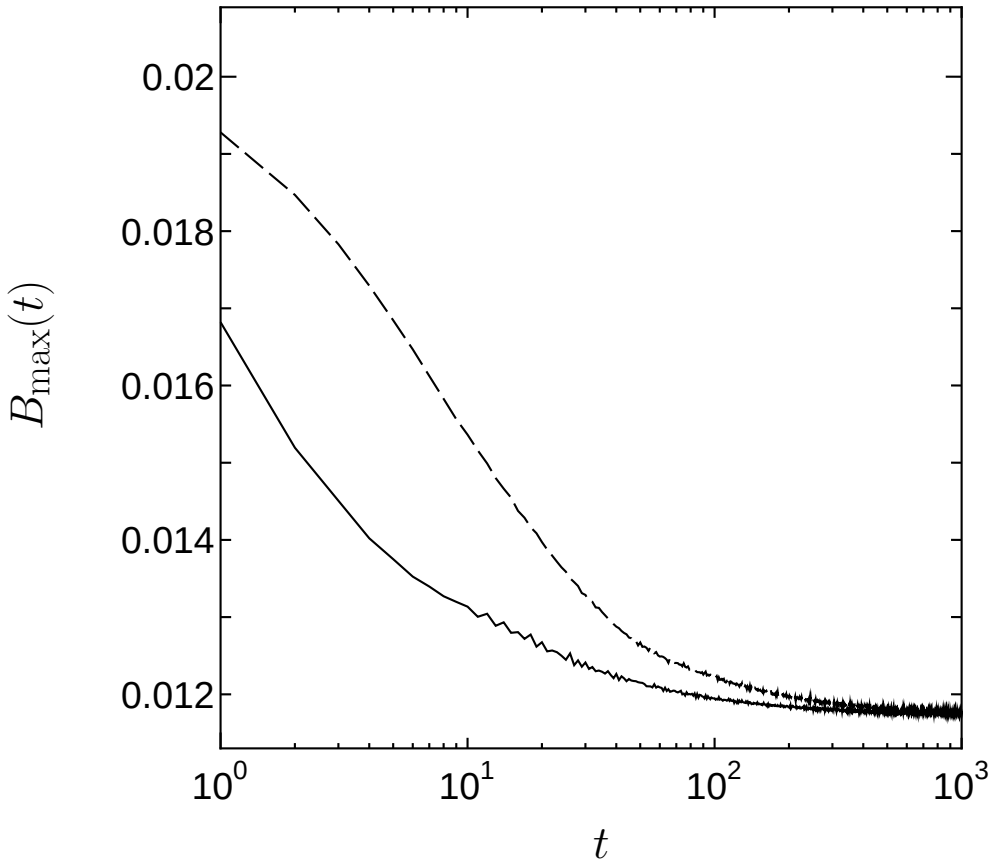

Fig. 4: Comparison between $B_{\max}(t)$ as a function of t for algorithm 2 ($a = 100$, dashed line) and algorithm 3 ($a(t) = 5 + t$, solid line) averaged over 100 samples is shown. Convergence is faster with algorithm 3 than algorithm 2 and the limit values are similar to each other.

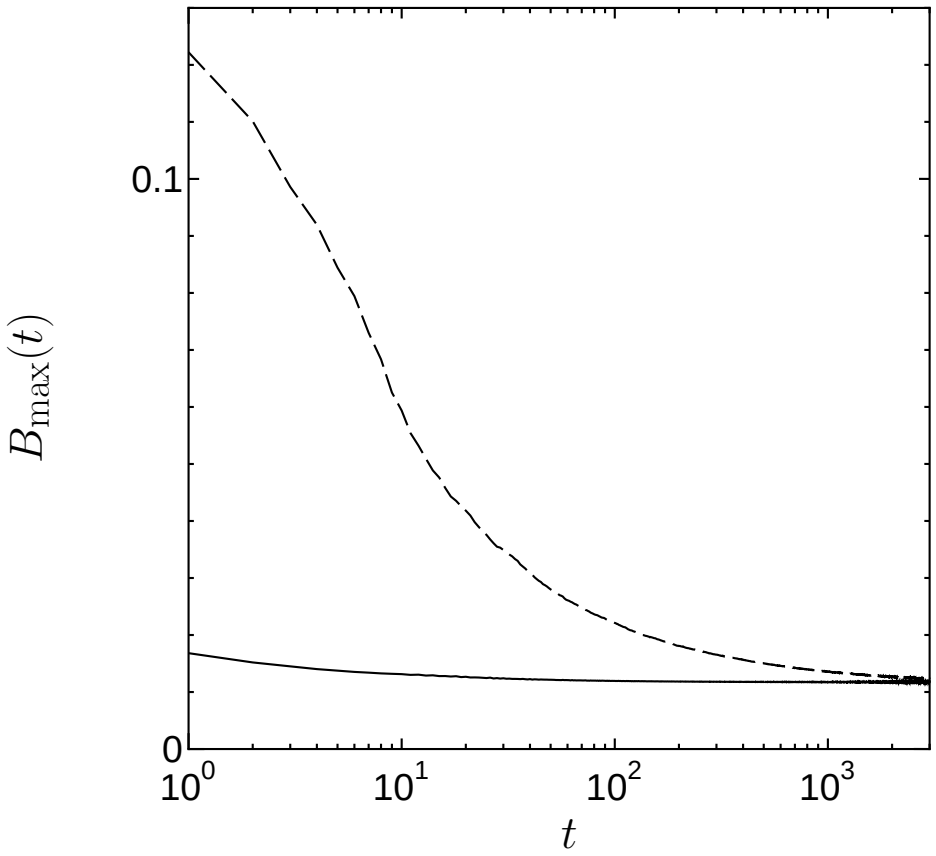

Fig. 5: $B_{\max}(t)$ as a function of t averaged over 100 samples. The results by the NOPS (solid line) are compared with those by the OPS (dashed line).

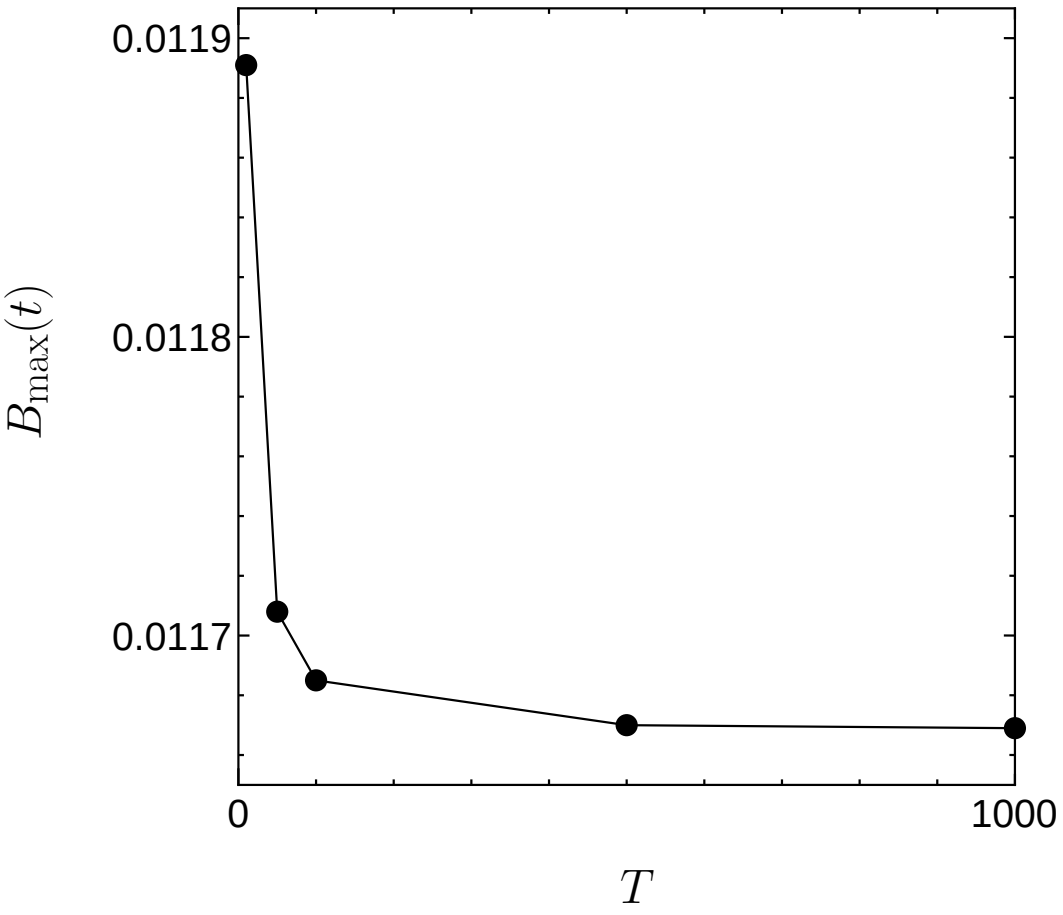

Fig. 6: $(B_{\max}(t))_{\min}$ by the NOPS with $N = 500$ and $\langle k \rangle = 8$ for the BA model as a function of T averaged over 100 samples.

3. The iteration is stopped if $(B_{\max}(t))_{\min}$ is not renewed for T steps.

We obtain $(B_{\max}(t))_{\min}$ using the NOPS in order to estimate T for convergence to adequately low $(B_{\max}(t))_{\min}$. We show the result with the average number of degrees $\langle k \rangle = 8$ in Fig.6 for $N = 500$. From the figure, the $(B_{\max}(t))_{\min}$ is converged to a sufficiently low value at $T = 500$ for $N = 500$. The results for other values of $\langle k \rangle$ and $N \in [300, 1000]$ are similar to those in the figure.

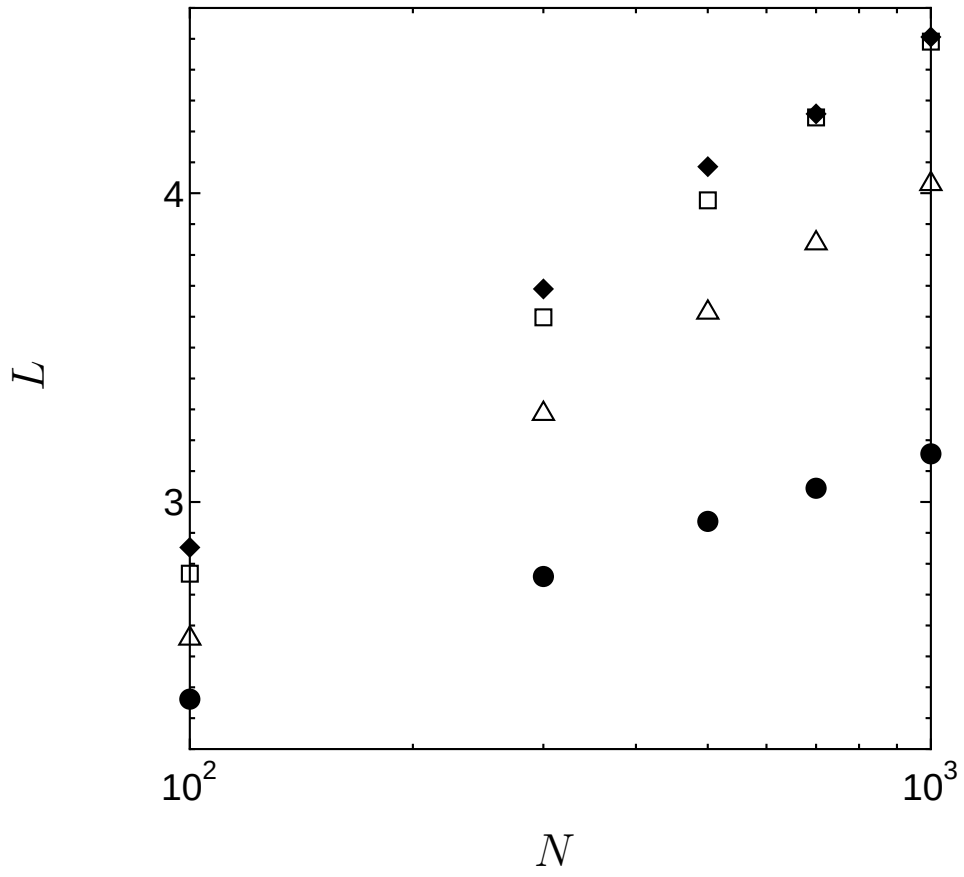

Fig. 7: We compare the average path length as a function of N for $\langle k \rangle = 8$ by four algorithms, the SPS (circles), DEPS (triangles), OPS (diamonds) and our algorithm (squares) averaged over 10 networks.

4.4. Average path length, average of packet traveling time and congestion index

Next, we show the average path length, average of packet traveling time, and congestion index $\eta(\lambda)$ on the network using the efficient paths with the weights optimized by the NOPS to investigate the efficiency of this algorithm for T=500 in the following figures. Packet traveling time is the number of steps from generation to arrival at the destination of a packet.

In Fig.7, we show the average path length L using four strategies, the NOPS, OPS, DEPS, and SPS as a function of the total number of nodes N. This figure shows that, in the range $N \in [100, 1000]$, $L \sim \log N$ is maintained

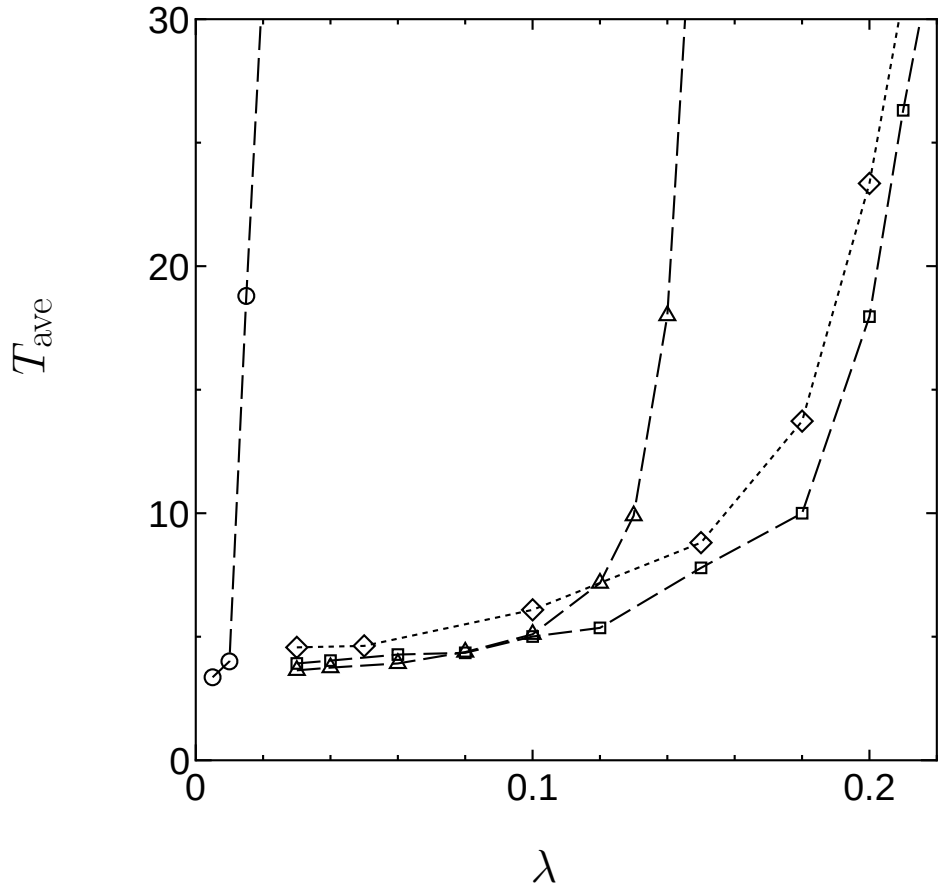

Fig. 8: We compare the average traveling time of a packet $T_{\rm ave}$ as a function of λ for $N = 500$ and $\langle k \rangle = 8$ for the SPS (circles), DEPS (triangles), OPS (diamonds) and our algorithm (squares) averaged over 30 networks and 500 steps.

by the NOPS though L with the use of the NOPS is a little smaller than that with the OPS and longer than the DEPS and SPS. In particular, L for the NOPS is approximately 1.5 times larger than that for the SPS.

In Fig.8, we show the average traveling time of a packet $T_{\rm ave}$ as a function of packet generation rate λ. From this figure, since, for the NOPS, $\lambda_{\rm c}$ where $T_{\rm ave}$ begins to increases intensely is much larger than those for the DEPS and SPS, it is found that the NOPS is proper for the tolerance of congestion although the packet traveling time by the NOPS is a little longer than those with the DEPS and SPS in the free flow state. The results with the OPS are slightly larger than those with the NOPS.

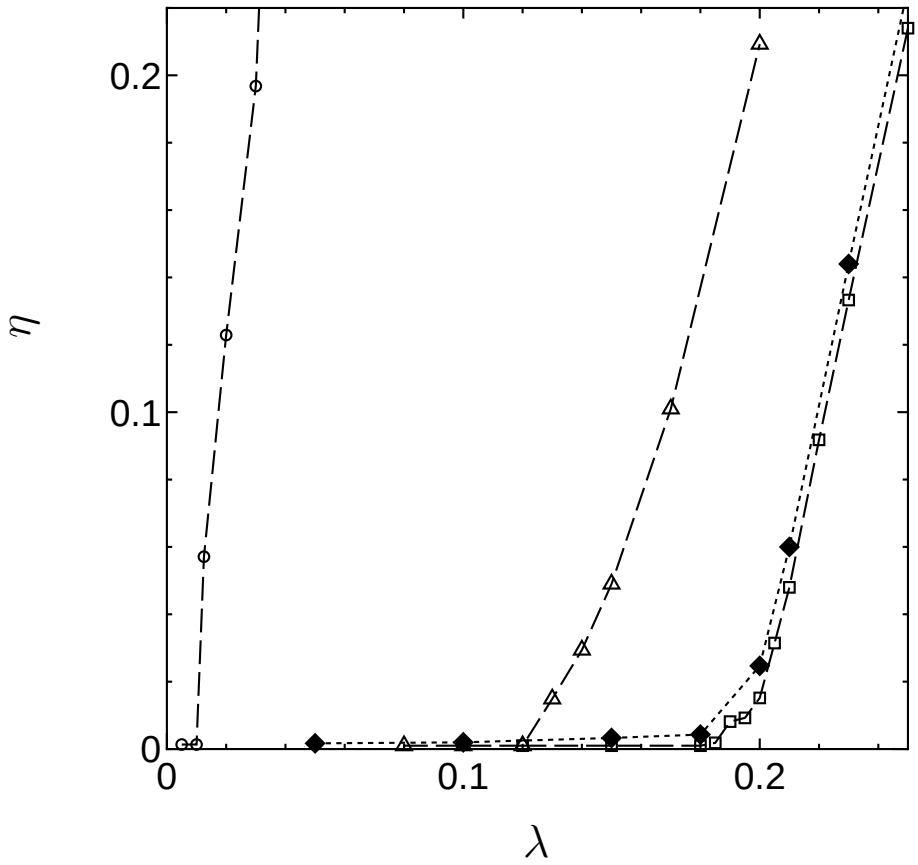

Fig. 9: We depict $\eta(\lambda)$ on the NOPS (squares), OPS (diamonds), DEPS (triangles) and SPS (circles), as a function of λ, $\langle k \rangle = 8$ and $N = 500$ averaged over 30 networks and 500steps.

The congestion index $\eta(\lambda)$ by the NOPS, OPS, DEPS and SPS is shown in Fig.9 as a function of the packet generation rate λ. From Fig.9, it is found that the NOPS is much more efficient than the DEPS, since λ_c by the NOPS is larger than that by the DEPS and SPS. $\eta(\lambda)$ with the OPS is similar to that with the NOPS.

The values of λ_c with the NOPS, OPS, DEPS and SPS, which are estimated from Figs.(8) and (9) are close to 0.17, 0.16, 0.097 and 0.014, respectively, calculated by Eq.(3).

4.5. Calculation time and the limit value of $(B_{\max}(t))_{\min}$

In this subsection, we investigate the calculation time required for determining the optimal weights and limit value of $(B_{\max}(t))_{\min}$ by the OPS and our algorithm.

We require $O(N^2 \log N)$ time in computing betweenness [23], $O(N^2)$ time in recomputing paths for both the NOPS and OPS, and in renewing the weight $O(1)$ time for the OPS and $O(N)$ time for the NOPS. As a result, the computational complexity required for one iteration is $O(N^2 \log N)$ for both the OPS and NOPS. On the other hand, the required iteration step to obtain limit $(B_{\max}(t))_{\min}$ is $O(N)$ for the OPS.[5] Therefore, for comparison of the iteration steps required for the two algorithms we need to estimate the required iteration step for the NOPS.

The iteration number $T_{\rm stop}$ in which the algorithm stops for the NOPS at T=500 is about 1000 and seems not to depend on the network size N in the range of $[300, 1000]$. The iteration number required for the OPS is $O(N)$ and that for the NOPS seems to be $O(1)$ in the range of $N \in [300, 1000]$. Further investigation is, however, needed for confirmation of this point.

We find from simulation that limit values of $(B_{\max}(t))_{\min}$ with the OPS and our algorithm are similar to each other if the algorithms are run for sufficient steps.

5. Conclusions

In Sec.4, we consider the heuristic algorithm which tunes the weight of every node by using the information of betweenness in every iteration step. Danila *et al.* have proposed the heuristic algorithms for convergence. There is, however, the problem that it takes quite long steps for convergence in the algorithm. We propose two points of improvement to solve this problem. We use the DEPS as the initial weights and change simultaneously the weights of several nodes in each step. As a result, it is found that the NOPS have the best performance when we change the weights of comparatively many nodes with large betweennesses in the early stage and change, in the late stage, only the weights of nodes with betweennesses which are extremely close to the maximum betweenness. We can reduce convergence time by using this algorithm while the low limit value of $B_{\max}$ is maintained. We mainly confirm the superiority of our algorithm by computer simulation for $N = 500$ and $\langle k \rangle = 8$.

We set the same capacity of delivering packets for each node in this paper. However it is known that there is a positive correlation between the degree and the capacity of delivering packets in real networks. In addition, if we model the distance between nodes and the delivering capacity of links in detail, we should set the different weight of each link [19]. We hope to investigate the efficiency of our routing strategy in the near future, taking these properties into account.

References

[1] D. J. Watts, S. H. Strogatz, Nature 286 (1998) 440.

[2] A. L. Barabási, R. Albert, Science 286 (1999) 509.

[3] M. E. J. Newman, SIAM Rev. 45 (2003) 167.

[4] K. I. Goh, E. Oh, B. Kahng, D. Kim, Phys. Rev. E 67 (2003) 017101.

[5] B. Danila, Y. Yu, J. A. Marsh, K. E. Bassler, Phys. Rev. E 74 (2006) 046106.

[6] B. Danila, J. A. Marsh, K. E. Bassler, CHAOS 17 (2007) 026102.

[7] Y. Yu, B. Danila, J. A. Marsh, K. E. Bassler, EPL 79 (2007) 48004.

[8] B. Danila, Y. Yu, S. Earl, J. A. Marsh, Z.Torocakai, K. E. Bassler, Phys. Rev. E 74 (2006) 046114.

[9] B. Danila, Y. San, K. E. Bassler, Phys. Rev. E 80 (2009) 066116.

[10] R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, A.Arenas, Phys. Rev. Lett. 89 (2001) 016132.

[11] R. Guimerà, A. Arenas, A. Díaz-Guilera, F. Giralt, Phys. Rev. E 89 (2002) 2478701.

[12] R. Guimerà, S. Mossa, A. Turtschi, L. A. N. Amaral, Proc. Nat, Acad. Sci. 102 (2005) 7794.

[13] I. Glauche, W. Krause, R. Sollacher, M. Greiner, Physica A 325 (2003) 577.

[14] W. Krause, I. Glauche, R. Sollacher, M. Greiner, Physica A 338 (2004) 633.

[15] I. Glauche, W. Krause, R. Sollacher, M. Greiner, Physica A 341 (2004) 677.

[16] W. Krause, M. Scholz, M. Greiner, Physica A 361 (2006) 707.

[17] G. Yan, T. Zhou, B. Hu, Z. Fu, and B. Wang, Phys. Rev. E **73** (2006) 046108.

[18] S. Sreenivasan, R. Cohen, E. López, Z. Toroczkai and H. E. Stanley, Phys. Rev. E **75** (2007) 036105.

[19] H. P. Thadakamalla, R. Albert, and S. R. T. Kumara, Phys. Rev. E, **72** (2005) 066128.

[20] M. E. J. Newman, Phys.Rev. E 64 (2001) 016132.

[21] M. E. J. Newman, M. Girvan, Phys. Rev. E 69 (2004) 026113.

[22] T. N. Buiand and C. Jones, Inf. Process. lett. **42** (1992) 153.

[23] U. Brandes, J. Math. Soc. **25** (2001) 163.